

Optimization-based smoothing algorithm for triangle meshes over arbitrarily shaped domains

D. Aubram*

Chair of Soil Mechanics and Geotechnical Engineering, TU Berlin, Germany

Abstract

This paper describes a node relocation algorithm based on nonlinear optimization which delivers excellent results for both unstructured and structured plane triangle meshes over convex as well as non-convex domains with high curvature. The local optimization scheme is a damped Newton's method in which the gradient and Hessian of the objective function are evaluated exactly. The algorithm has been developed in order to continuously rezone the mesh in arbitrary Lagrangian-Eulerian (ALE) methods for large deformation penetration problems, but it is also suitable for initial mesh improvement. Numerical examples highlight the capabilities of the algorithm.

Keywords: mesh; triangle; smoothing; optimization; large deformation; arbitrary Lagrangian-Eulerian

1 Introduction

In every mesh-based numerical method the convergence of the solution algorithms and the accuracy of the solution results depend on the quality of the mesh. Mesh improvement usually becomes necessary, at least in postprocessing the originally generated mesh. Mesh improvement is often initiated if a quality measure drops below a certain value specified by the user. Physical quality measures are employed in the adaptive numerical methods for initial boundary value problems. Geometric quality measures, including the size, aspect ratio, and skew of a mesh element, can be evaluated independently of the physical solution and usually at lower computational costs [1, 2, 3, 4].

*Correspondence to: Dr.-Ing. Daniel Aubram, Chair of Soil Mechanics and Geotechnical Engineering, Technische Universität Berlin (TU Berlin), Secr. TIB1-B7, Gustav-Meyer-Allee 25, D-13355 Berlin, Germany. Tel.: +49 (0)30 31472349; fax: +49 (0)30 31472343; E-mail: daniel.aubram@tu-berlin.de

The quality improvement of a mesh can be governed by quality evolution and is done by repeated application of appropriate tools. Smoothing is a tool intended to improve mesh quality by node relocation. It represents a class of homeomorphic maps between meshes which keep the connectivity of the original mesh unchanged. Smoothing plays a crucial role in the arbitrary Lagrangian-Eulerian (ALE) methods used for large deformation problems with interfaces in computational solid and fluid dynamics [5, 6, 7, 8, 9, 10, 11]; see [12, 13] for a review. ALE methods combine the advantages of the purely Lagrangian and purely Eulerian approaches. The computational mesh is not fixed but can move independent of the material at an arbitrary velocity prescribed by the smoothing scheme.

Since ALE methods must frequently relocate the mesh nodes when advancing solution of the considered problem in time, an essential requirement for the smoothing scheme is efficiency with respect to computational costs. Another requirement closely connected with efficiency is locality, that means to process only a set of flagged nodes which may vary between the time steps. When using local procedures, attention must be drawn to the strategy in order to globally smooth the mesh. This holds for all local improvement tools. Any improved mesh entity may deteriorate the quality of neighboring entities. The third requirement imposed on a smoothing algorithm is stability. A stable smoothing algorithm will not distort a mesh any more than it is currently distorted [7]. For an algorithm to be reliable, this should be independent of the domain's shape.

Automatic mesh smoothing procedures which are not governed by quality evolution are called direct or heuristic smoothing algorithms. Examples include Laplacian smoothing [4], smoothing by weighted averaging [14], and Giuliani's method [15]. These methods provide closed-form expressions for the new node location which is supposed to smooth the associated ball of elements sharing that node. Even though these methods are computationally attractive, they cannot ensure quality improvement for arbitrarily shaped domains. As will be shown later, the heuristic smoothing algorithms proposed in [14] and [15] fail on a non-convexly distorted mesh.

Another class of algorithms is referred to as physically-based smoothing. In these algorithms, physical properties are assigned to the mesh entities and then a specific initial boundary value problem is solved over a dummy time step in order to determine the nodal displacements. Examples of physically-based smoothing methods are reported in [16, 8]. The success of such a procedure, however, is by pure chance. It cannot be ensured that any mesh processed is not worse than before, i.e. that the smoothing scheme is stable.

The drawbacks of heuristic and physically-based smoothing techniques when dealing with non-convex meshes can be avoided if the new node positions are determined through an optimization process. In contrast to the other two approaches, optimization-based smoothing algorithms are governed by geometric quality evolution using an objective function whose minimum is associated with a properly smoothed mesh, or a part of it. Early references include [17, 18] which are con-

cerned with the global optimization of two-dimensional structured grids. One of the first mesh smoothing algorithms that use principles of local optimization is developed in [19]. Several refinements of the local approach and its generalization to unstructured three-dimensional meshes are provided, for example, in [20, 21]. Development continued up to the present, with focuses on unstructured quadrilateral meshes [22], unstructured triangle meshes [23], structured quadrilateral meshes [24], and general unstructured polyhedral meshes [25]. These algorithms share the basic structure of all optimization procedures [26, sec. 1.5] but differ in the methods to determine the descent direction and step size, and particularly in the objective function.

The remainder of this paper is concerned with the development of an optimization-based smoothing algorithm for two-dimensional triangle meshes, which shall be referred to as the OSMOT (Optimization-based SMOothing of Triangle meshes) algorithm. The mesh can be an originally generated mesh but the main objective of this research is to efficiently smooth distorted meshes over non-convex domains arising in ALE finite element simulations of penetration problems. Section 2 describes the procedure to globally improve the mesh. The global algorithm encloses local algorithms to smooth the boundary mesh and the internal mesh which are outlined in Sections 3 and 4, respectively. Extensions of the algorithm are discussed in Section 5. The numerical examples presented in Section 6 highlight that the new algorithm delivers excellent results for both unstructured and structured plane triangle meshes over convex as well as non-convex domains with high curvature. The paper closes with some concluding remarks in Section 7.

2 Global algorithm

2.1 General setup and initialization

Let \mathcal{M} be a two-dimensional triangle mesh in the Euclidian space $\mathcal{S} = \mathbb{R}^2$ and let $\mathcal{N}(\mathcal{M})$ be the set of all nodes in the mesh. The position vector of a node $P_0 \in \mathcal{N}(\mathcal{M})$ is given by $\mathbf{x}_0 = (x_0, y_0)^T \in \mathbb{R}^2$ with respect to the canonical basis of \mathbb{R}^2 . The superscribed T denotes the transpose of a matrix. Some frequently used geometric primitives of triangles are compiled in Appendix A

The current procedure assumes that all nodes of the mesh are allowed to be moved, except for the boundary nodes that essentially define the shape of the meshed domain. The set of internal nodes lying in the interior of the mesh is denoted by $\mathcal{N}_{\text{int}} \subset \mathcal{N}$. The non-movable boundary nodes divide the boundary into a number of n_{bnd} distinct sub-boundaries, and the set of all movable nodes of the j -th sub-boundary is denoted by \mathcal{N}_{∂}^j , with $j \in \{1, \dots, n_{\text{bnd}}\}$.

The algorithms intended to smooth the interior of a mesh generally can not directly be applied to the boundary mesh. In most cases the quality improvement of a

distorted boundary mesh can be achieved by simple heuristic procedures. Weighted averaging [14] is used here, whereas a new optimization-based procedure is applied to smooth the internal mesh. These are local algorithms in order to render the global improvement of the whole mesh more effective.

The implemented local algorithms require additional topological information. In particular, the local algorithm for internal nodes works on the ball of elements associated with some node $P_0 \in \mathcal{N}$. A ball is the disjoint union $\mathcal{B}(P_0) \stackrel{\text{def}}{=} \bigcup_{n_{\text{el}}} \Delta(P_0)$ of all n_{el} elements Δ in a mesh sharing P_0 , the vertex of the ball. Locally, the numbering of the nodes in each triangle element of the ball is reordered such that the signed area of the element (A1) is positive and the location of the local node $0 \in \Delta$ in $\mathcal{S} = \mathbb{R}^2$ coincides with that of P_0 . The reordering of the local node numbers ensures that for each element $\Delta \subset \mathcal{B}$ the vertex of the ball can be addressed by $\mathbf{x}_0 \in \mathbb{R}^2$, the position vector of the local node $0 \in \Delta$.

2.2 Selection of the nodes to be moved

Smoothing is initiated if at least one mesh element fails a quality check. Stated loosely, a geometrically high quality mesh is made up of more or less equal-sized elements with low distortion. The two main groups of geometric quality measures are accordingly referred to as size measures and shape measures. The group of shape measures includes measures for the aspect ratio and skew of an element [3].

For simplicial elements a size measure can be established by taking the ratio of a reference radius R_{ref} and the circumcircle radius R :

$$Q_1 \stackrel{\text{def}}{=} \frac{R_{\text{ref}}}{R} . \quad (1)$$

However, Q_1 is a fair size measure only if the physical element is almost regular, since R can be finite even if element volume is not (degenerate element).

A widely-used and versatile shape measure for simplicial elements because it covers aspect ratio and skew is the normalized radius ratio of the incircle and circumcircle [1, 2, 23]:

$$Q_2 \stackrel{\text{def}}{=} m \frac{r}{R} \in [0, 1] . \quad (2)$$

The normalization factor m is the dimension of the simplex, with $m = 2$ (triangle) or $m = 3$ (tetrahedron). A simplicial element is equilateral if $Q_2 = 1$, and has zero volume if $Q_2 = 0$.

The geometric quality Q_Δ of each mesh element $\Delta \in \mathcal{M}$ is compared with a minimal acceptable quality Q_{min} . The nodes of the elements that fail the quality check are flagged. Hence, the set of flagged nodes intended for relocation is given by

$$\mathcal{N}' \stackrel{\text{def}}{=} \{P \in \mathcal{N}(\mathcal{M}) \mid P \in \mathcal{N}(\Delta) \text{ and } \Delta \in \mathcal{M} \text{ and } Q_\Delta < Q_{\text{min}}\} , \quad (3)$$

with $\mathcal{N}' \subset \mathcal{N}$. Computational costs of the global algorithm can be considerably reduced by processing only the flagged nodes by the local smoothing algorithms. Because the total number of boundary nodes is not very large in two-dimensional meshes, however, it would be adequate to relocate all movable boundary nodes without any prior quality check.

2.3 Global iteration

The globally improved mesh is obtained by looping over the flagged nodes of the mesh repeatedly. Hence, smoothing of the whole mesh is achieved in an iterative fashion. Alg. 1 provides the pseudocode of the entire procedure.

Algorithm 1: Global mesh smoothing.

Input: triangle mesh \mathcal{M} , locations of the nodes $\mathcal{N}(\mathcal{M})$

Output: smoothed mesh

- 1 initialize $i = 0$, specify i_{\max} and Q_{\min} ;
 - 2 specify set of movable nodes \mathcal{N}_{∂}^j for every sub-boundary $j \in \{1, \dots, n_{\text{bnd}}\}$;
 - 3 specify set of movable internal nodes \mathcal{N}_{int} ;
 - 4 **foreach** $P_0 \in \mathcal{N}_{\partial}^j$ with $j \in \{1, \dots, n_{\text{bnd}}\}$ **do**
 - 5 \lfloor determine neighboring nodes P_1, P_2 ;
 - 6 **foreach** $P_0 \in \mathcal{N}_{\text{int}}$ **do**
 - 7 \lfloor determine ball of elements $\mathcal{B}(P_0) = \bigcup_{n_{\text{el}}} \Delta(P_0)$;
 - 8 loop elements and evaluate element quality Q_{Δ} ;
 - 9 **if** $Q_{\Delta} < Q_{\min}$ **then** flag nodes of element (set of all flagged nodes is \mathcal{N}');
 - 10 **while** global iteration step $i \leq i_{\max}$ **do**
 - 11 **foreach** $P_0 \in \mathcal{N}_{\partial}^j$ with $j \in \{1, \dots, n_{\text{bnd}}\}$ **do**
 - 12 \lfloor determine new location to smooth boundary mesh (Alg. 2);
 - 13 **foreach** $P_0 \in (\mathcal{N}' \cap \mathcal{N}_{\text{int}})$ **do**
 - 14 \lfloor smooth ball $\mathcal{B}(P_0)$ of internal mesh (Alg. 3);
 - 15 $i \leftarrow i + 1$;
-

3 Local algorithm for boundary nodes

The heuristic algorithm of Aymone *et al.* [14] efficiently smoothes mesh boundaries. The boundary node $P_0 \in \mathcal{N}_{\partial}^j$ intended for relocation has two neighbors, P_1 and P_2 , which also belong to the boundary. To relocate P_0 , one assumes that the three points lie on a sufficiently smooth curve

$$\begin{aligned} c : [-1, 1] &\rightarrow \mathcal{S} = \mathbb{R}^2 \\ \xi &\mapsto c(\xi), \exists c^{-1}, \end{aligned} \tag{4}$$

with $c(-1) = P_1$, $c(0) = P_0$, and $c(1) = P_2$. The position vector of a point $c(\xi)$ is $\mathbf{c}(\xi) \stackrel{\text{def}}{=} \mathbf{x}(c(\xi))$. Now the boundary curve through P_1, P_0, P_2 considered in [14] is a polynomial of degree two such that $\mathbf{c}(\xi)$, with $\xi \in [-1, 1]$, has the exact representation

$$\mathbf{c}(\xi) = \sum_{k=0}^2 N_k(\xi) \mathbf{x}_k. \quad (5)$$

Here \mathbf{x}_k , $k \in \{0, 1, 2\}$, are the position vectors of P_k in \mathcal{S} , and N_k are quadratic interpolation functions for \mathbf{c} and P_k having the particular form

$$N_0(\xi) \stackrel{\text{def}}{=} 1 - \xi^2, \quad N_1(\xi) \stackrel{\text{def}}{=} \frac{1}{2}(\xi^2 - \xi), \quad \text{and} \quad N_2(\xi) \stackrel{\text{def}}{=} \frac{1}{2}(\xi^2 + \xi). \quad (6)$$

A straightforward quality measure for the local boundary mesh formed by P_1, P_0, P_2 is

$$Q_{\text{bnd}} \stackrel{\text{def}}{=} \frac{\min(\|\mathbf{x}_1 - \mathbf{x}_0\|, \|\mathbf{x}_2 - \mathbf{x}_0\|)}{\max(\|\mathbf{x}_1 - \mathbf{x}_0\|, \|\mathbf{x}_2 - \mathbf{x}_0\|)} \in [0, 1], \quad (7)$$

with $\|\mathbf{x}_k - \mathbf{x}_l\| = \sqrt{(x_k - x_l)^2 + (y_k - y_l)^2}$ and $k, l \in \{0, 1, 2\}$. $Q_{\text{bnd}} = 1$ means that the location of the node P_0 equalizes the distances (best quality). In [14], weighted averaging is applied to determine a natural coordinate $\xi'_0 \in [-1, 1]$ of P_0 that smoothes the boundary curve. By using the distances $\|\mathbf{x}_1 - \mathbf{x}_0\|$ and $\|\mathbf{x}_2 - \mathbf{x}_0\|$ as weights one arrives at

$$\xi'_0 = \frac{\|\mathbf{x}_2 - \mathbf{x}_0\| - \|\mathbf{x}_1 - \mathbf{x}_0\|}{\|\mathbf{x}_1 - \mathbf{x}_0\| + \|\mathbf{x}_2 - \mathbf{x}_0\|}. \quad (8)$$

The new position vector \mathbf{x}'_0 of P_0 that smoothes the boundary curve can be obtained from (5) by using the coordinate $\xi = \xi'_0$, so that $\mathbf{x}'_0 = \mathbf{c}(\xi'_0)$. The procedure is summarized in Alg. 2.

Algorithm 2: Local smoothing for boundary nodes.

Input: neighboring nodes P_1, P_2 of every $P_0 \in \mathcal{N}_\partial^j$

Output: smoothed position of $P_0 \in \mathcal{N}_\partial^j$

- 1 read locations of nodes $\mathbf{x}_0 = \mathbf{x}(P_0)$, $\mathbf{x}_1 = \mathbf{x}(P_1)$, and $\mathbf{x}_2 = \mathbf{x}(P_2)$;
 - 2 compute distances $\|\mathbf{x}_1 - \mathbf{x}_0\|$ and $\|\mathbf{x}_2 - \mathbf{x}_0\|$;
 - 3 natural coordinate to equalize distances is $\xi'_0 = \frac{\|\mathbf{x}_2 - \mathbf{x}_0\| - \|\mathbf{x}_1 - \mathbf{x}_0\|}{\|\mathbf{x}_1 - \mathbf{x}_0\| + \|\mathbf{x}_2 - \mathbf{x}_0\|}$;
 - 4 location of P_0 smoothing the local boundary mesh is $\mathbf{x}'_0 = \sum_{k=0}^2 N_k(\xi'_0) \mathbf{x}_k$, with N_k given by (6);
-

4 Local optimization algorithm for internal nodes

4.1 General remarks

Finding the best location of mesh nodes in terms of geometric element quality constitutes an optimization problem which can be solved using optimization theory

[26, 27]. Let $\mathcal{X} \subset \mathbb{R}^m$ be a feasible region, $\mathbf{x} \in \mathcal{X}$, and $f : \mathcal{X} \rightarrow \mathbb{R}$ a function. The general optimization problem can then be stated as follows:

$$\text{minimize } f(\mathbf{x}) \quad \text{subject to } \mathbf{x} \in \mathcal{X}.$$

The function f is called the objective function, and the optimization problem is called unconstrained if $\mathcal{X} = \mathbb{R}^m$. In the remainder of this paper, the objective function is assumed to be twice continuously differentiable in the Fréchet-sense on \mathcal{X} , i.e. of class C^2 such that its gradient $\nabla f(\mathbf{x}) \in \mathbb{R}^m$ and its Hessian $\mathbf{H}_f(\mathbf{x}) \in \mathbb{R}^{m \times m}$ at point $\mathbf{x} \in \mathcal{X}$ do exist.

Determination of global minimizers is challenging. However, it is usually sufficient to determine a local minimizer and to iterate the global minimum. In this context the following first- and second-order conditions are of fundamental importance. Proofs can be found in [26, sec. 1.4].

Theorem 1 (i) *If $f : \mathcal{X} \rightarrow \mathbb{R}$ is continuously differentiable on $\mathcal{X} \subset \mathbb{R}^m$ and $\mathbf{x}' \in \arg \min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$ is a local minimizer of f , then*

$$\nabla f(\mathbf{x}') = \mathbf{0}.$$

(ii) *If $\mathbf{x}' \in \mathcal{X}$ is a local minimizer of f , and f is of class C^2 on \mathcal{X} , then $\nabla f(\mathbf{x}') = \mathbf{0}$ and the Hessian $\mathbf{H}_f(\mathbf{x}')$ is positive semidefinite, i.e. $\mathbf{x}^T \mathbf{H}_f \mathbf{x} \geq 0$ for every $\mathbf{x} \in \mathbb{R}^m$ with $\mathbf{x} \neq \mathbf{0}$.*

(iii) *Let f be C^2 on \mathcal{X} , $\mathbf{x}' \in \mathcal{X}$, $\nabla f(\mathbf{x}') = \mathbf{0}$, and let $\mathbf{H}_f(\mathbf{x}')$ be positive definite such that $\mathbf{x}^T \mathbf{H}_f \mathbf{x} > 0$ for every $\mathbf{x} \in \mathbb{R}^m$ with $\mathbf{x} \neq \mathbf{0}$, then \mathbf{x}' is a strict local minimizer.*

Finding a local minimizer to solve the optimization problem usually is an iterative procedure. Let $\mathcal{J} \in \mathbb{N}$ be an index set and $j, j+1 \in \mathcal{J}$. For a given \mathbf{x}^j , the iterative procedure takes the form

$$\mathbf{x}^{j+1} = \mathbf{x}^j + \lambda^j \mathbf{d}^j, \quad (9)$$

where $\mathbf{d} \in \mathbb{R}^m$ is a descent direction of f at \mathbf{x} satisfying $(\nabla f(\mathbf{x}))^T \mathbf{d} < 0$. Once a starting point $\mathbf{x}^{j=0}$, a step size $\lambda^{j=0} > 0$, and a tolerance $\varepsilon > 0$ have been specified, a termination criterion of the form

$$\|\nabla f(\mathbf{x}^j)\| < \varepsilon \quad (10)$$

is checked. If this criterion is met, then $\mathbf{x}^j \approx \mathbf{x}'$ is an approximate minimizer of f . If the criterion is not met, the descent direction \mathbf{d}^j supposed to point to the minimum of the objective function is determined by some method. Thereafter, a so-called line search is carried out in order to determine the step size λ^j satisfying

$$f(\mathbf{x}^j + \lambda^j \mathbf{d}^j) < f(\mathbf{x}^j). \quad (11)$$

An effective step size rule is highly desirable to ensure a sufficient decrease in the objective function. The iterative procedure continues with the repeated evaluation of the termination criterion using $\mathbf{x}^{j+1} = \mathbf{x}^j + \lambda^j \mathbf{d}^j$.

4.2 Objective function

The choice of an objective function is crucial to the success of optimization-based mesh smoothing. It must be composed of geometric quality measures to ensure that the optimization is governed by quality evolution. The class of local objective functions for triangle meshes considered here takes the form [23]

$$W(\mathbf{x}_0) \stackrel{\text{def}}{=} \sum_{n_{\text{el}}} w(\mathbf{x}_0), \quad \text{with} \quad w(\mathbf{x}_0) \stackrel{\text{def}}{=} \left(\frac{R(\mathbf{x}_0)}{R_{\text{ref}}} \right)^\beta \left(\frac{R(\mathbf{x}_0)}{r(\mathbf{x}_0)} \right)^\gamma. \quad (12)$$

n_{el} is the number of triangles in the ball $\mathcal{B}(P_0) = \bigcup_{n_{\text{el}}} \triangle(P_0)$ associated with the internal node $P_0 \in \mathcal{N}'$ whose position vector is \mathbf{x}_0 , β and γ are constant positive weighting exponents, and $R_{\text{ref}} > 0$ is a constant reference radius.

The class of local objective functions defined through (12) takes into account the size quality measure (1) and the shape quality measure (2). The weighting exponents β and γ control the domination of the worst element. For example, if γ is large and β is moderate, then the most distorted element contributes more to the sum than a too large element or any of the remaining elements. For the purpose of the present work, the values $\beta = 1.0$, $\gamma = 3.0$, and $R_{\text{ref}} = 1.0$ have been assigned to all elements in a mesh; see also [23].

4.3 Descent direction

The first-order necessary condition (Theorem 1(i)) in conjunction with (12) defines a homogeneous system of generally nonlinear algebraic equations, $\nabla W(\mathbf{x}'_0) = \mathbf{0}$, whose solution is $\mathbf{x}'_0 \in \arg \min_{\mathbf{x}_0 \in \mathcal{X}} W(\mathbf{x}_0)$, the new location of the vertex of the ball $\mathcal{B}(P_0)$. The solution can be approximated by Newton's method. Let \mathbf{x}_0^j be a close-enough guess of the local minimizer. For $W : \mathcal{X} \rightarrow \mathbb{R}$ being a C^2 -function in the neighborhood of \mathbf{x}_0^j , the linearization of $\nabla W(\mathbf{x}'_0) = \mathbf{0}$ about \mathbf{x}_0^j leads to

$$\nabla W(\mathbf{x}_0^j) + \mathbf{H}_W(\mathbf{x}_0^j) \cdot (\mathbf{x}'_0 - \mathbf{x}_0^j) \approx \mathbf{0}. \quad (13)$$

Provided that the gradient ∇W is a sufficiently smooth function, then any guess \mathbf{x}_0^{j+1} for which $\nabla W(\mathbf{x}_0^j) + \mathbf{H}_W(\mathbf{x}_0^j) \cdot (\mathbf{x}_0^{j+1} - \mathbf{x}_0^j) = \mathbf{0}$ is a better approximation than \mathbf{x}_0^j . If \mathbf{H}_W is regular on $\mathcal{X} \subset \mathbb{R}^2$, this latter condition results in the iterative scheme

$$\mathbf{x}_0^{j+1} = \mathbf{x}_0^j - (\mathbf{H}_W^{-1} \nabla W)(\mathbf{x}_0^j), \quad \text{and} \quad \lim_{j \rightarrow \infty} \mathbf{x}_0^{j+1} = \mathbf{x}'_0. \quad (14)$$

By linearity,

$$\nabla W(\mathbf{x}_0^j) = \sum_{n_{\text{el}}} \nabla w(\mathbf{x}_0^j) \quad \text{and} \quad \mathbf{H}_W(\mathbf{x}_0^j) = \sum_{n_{\text{el}}} \mathbf{H}_w(\mathbf{x}_0^j). \quad (15)$$

Closed-form expressions for the components of the gradient $\nabla w(\mathbf{x}_0^j)$ and the Hessian $\mathbf{H}_w(\mathbf{x}_0^j)$ are available through (A3)–(A6); see Appendix B for a straightforward calculation.

By locality of Newton's method, the iterative scheme (14) converges only if the starting point $\mathbf{x}_0^{j=0}$ is a close-enough guess of the solution. When the starting point is far away from the solution it is not guaranteed that the Hessian is invertible and positive definite at every $\mathbf{x}_0^j \in \mathcal{X}$ and that Newton's direction, $-(\mathbf{H}_W^{-1} \nabla W)(\mathbf{x}_0^j)$, is indeed a descent direction satisfying

$$(\nabla W)^T \mathbf{H}_W^{-1} \nabla W > 0 . \quad (16)$$

In these cases solution may diverge. Even if the starting point is close to solution, the Hessian may still be non-positive definite such that no strict local minimizer of the objective function exists (cf. Theorem 1(iii)).

In order to ensure convergence at non-positive definite Hessians, a modified Newton's method is employed. The particular approach used in this work was suggested by Goldstein and Price [28]. It substitutes the steepest descent direction $-\nabla W(\mathbf{x}_0^j)$ instead of $-(\mathbf{H}_W^{-1} \nabla W)(\mathbf{x}_0^j)$ for \mathbf{d}^j in (14) whenever \mathbf{H}_W is not regular or positive definite at \mathbf{x}_0^j . The check for positive definiteness is done by the angle criterion [26]. To this end, define

$$\cos \theta^j \stackrel{\text{def}}{=} -\frac{(\nabla W(\mathbf{x}_0^j))^T \mathbf{d}^j}{\|\nabla W(\mathbf{x}_0^j)\| \|\mathbf{d}^j\|} \quad (17)$$

at the j -th iteration. If $\cos \theta^j > 0$ for $\mathbf{d}^j = -(\mathbf{H}_W^{-1} \nabla W)(\mathbf{x}_0^j)$, then Newton's direction is indeed a descent direction (\mathbf{H}_W is positive definite) and the iterative scheme converges. If, on the other hand, $\cos \theta^j \leq 0$ for $\mathbf{d}^j = -(\mathbf{H}_W^{-1} \nabla W)(\mathbf{x}_0^j)$, then $\mathbf{d}^j = -\nabla W(\mathbf{x}_0^j)$ is used as the descent direction, satisfying $\cos \theta^j > 0$ when substituted into (17) as long as \mathbf{x}_0^j is not a minimizer of W . This can be implemented as follows:

$$\mathbf{d}^j \stackrel{\text{def}}{=} \begin{cases} -\nabla W(\mathbf{x}_0^j), & \text{if } \det \mathbf{H}_W(\mathbf{x}_0^j) < \delta \text{ or if } \cos \theta^j < \eta, \\ -(\mathbf{H}_W^{-1} \nabla W)(\mathbf{x}_0^j), & \text{otherwise,} \end{cases} \quad (18)$$

where $\delta > 0$ and $\eta > 0$ are reasonable tolerances.

4.4 Line search and step size rule

It remains to determine the size of the steps with which the optimization procedure approaches the local minimum of the objective function. A too large step size may overshoot the minimum, whereas a tiny step size would decelerate the overall procedure. Therefore, it is a mandatory goal to let the program determine an appropriate size for every step by a line search.

Inexact line search is preferable from a computational viewpoint provided that there is an effective step size rule which gives a sufficient decrease in the objective function. One of such rules is the widely-used Armijo rule [29],

$$W(\mathbf{x}_0^j + \lambda^j \mathbf{d}^j) - W(\mathbf{x}_0^j) \leq \frac{1}{2} \lambda^j (\nabla W(\mathbf{x}_0^j))^T \mathbf{d}^j , \quad (19)$$

resulting in a so-called backtracking line search [26]. In a backtracking line search, for given $W(\mathbf{x}_0^j)$, $\nabla W(\mathbf{x}_0^j)$, \mathbf{d}^j , and $\lambda^{j=0} = 1.0$, the condition (19) is checked. If it is satisfied, then $\lambda^{j+1} = \lambda^j$ and $\mathbf{x}_0^{j+1} = \mathbf{x}_0^j + \lambda^j \mathbf{d}^j$. If the condition is not satisfied, the current guess of the minimizer is used in the next iteration and the step size is bisected, that is,

$$\mathbf{x}_0^{j+1} = \mathbf{x}_0^j \quad \text{and} \quad \lambda^{j+1} = \frac{\lambda^j}{2}, \quad (20)$$

respectively.

4.5 Optimization procedure

The entire local optimization procedure for internal nodes is provided by Alg. 3. The included tolerances have been chosen to $\varepsilon = 10^{-8}$, $\delta = 10^{-6}$, and $\eta = 0.05$ for all the numerical examples presented in Section 6. Note that the algorithm is independent of the specific objective function assigned to a ball of elements. It might be attractive to implement alternative functions for which evaluation of the gradient and Hessian is much cheaper or which better reflect the user's needs.

5 Extensions to the current algorithm

The current algorithm can be naturally extended to adaptive smoothing, to three dimensions, to higher-order elements, and to surface meshes. The purpose of the following section is to discuss these extensions.

The reference radius used in the objective function (12) is an attribute assigned to every element and defines its maximum acceptable size in the mesh. Specifying appropriate R_{ref} thus controls mesh grading during the optimization process. If the value of the reference radius is not specified by the user but *a posteriori* by element quality measures based on a numerical solution, then the optimization-based algorithm would account for mesh grading, leading to *r*-adaptive mesh improvement.

The proposed algorithm is based on a simplicial element type, hence it should have a natural extension to three dimensions if the triangles are replaced by tetrahedra. The local optimization algorithm running over the internal nodes has a three dimensional analog because all simplicial elements have a unique incircle and a unique circumcircle, whose radii can be substituted into the objective function (12). In 3d, however, exact evaluation of the gradient and Hessian of the related objective function would yield awfully lengthy expressions. Moreover, for the boundary nodes in 3d smoothing is much more complicated as one will have to deal with surface triangulation connected to 3d elements. A three-dimensional extension of the 2d algorithm presented in Section 3 is proposed in [14], but its suitability for simplicial meshes is not clear.

Algorithm 3: Local optimization-based smoothing for internal nodes.

Input: ball $\mathcal{B}(P_0)$ associated with every $P_0 \in (\mathcal{N}' \cap \mathcal{N}_{\text{int}})$

Output: smoothed ball

```

1 specify tolerances  $\varepsilon$ ,  $\delta$ , and  $\eta$ ;
2 initialize  $j = 0$ ,  $\mathbf{x}_0^{j=0} = \mathbf{x}(P_0)$ , and  $\lambda^{j=0} = 1.0$ ;
3 while damped Newton iteration step  $j \leq j_{\max}$  do
4   initialize  $W(\mathbf{x}_0^j) = 0$ ,  $W(\mathbf{x}_0^j + \lambda^j \mathbf{d}^j) = 0$ ,  $\nabla W(\mathbf{x}_0^j) = \mathbf{0}$ ,  $\mathbf{H}_W(\mathbf{x}_0^j) = \mathbf{0}$ ;
5   while element in the ball  $e \leq n_{\text{el}}$  do
6     read and store locations of nodes  $\mathbf{x}_0^j$ ,  $\mathbf{x}_1$ , and  $\mathbf{x}_2$ ;
7     compute element objective function  $w(\mathbf{x}_0^j)$ ;
8     compute  $\nabla w(\mathbf{x}_0^j)$  and  $\mathbf{H}_w(\mathbf{x}_0^j)$  (B);
9      $W(\mathbf{x}_0^j) \leftarrow W(\mathbf{x}_0^j) + w(\mathbf{x}_0^j)$ ;
10     $\nabla W(\mathbf{x}_0^j) \leftarrow \nabla W(\mathbf{x}_0^j) + \nabla w(\mathbf{x}_0^j)$ ;
11     $\mathbf{H}_W(\mathbf{x}_0^j) \leftarrow \mathbf{H}_W(\mathbf{x}_0^j) + \mathbf{H}_w(\mathbf{x}_0^j)$ ;
12  if  $\|\nabla W(\mathbf{x}_0^j)\| < \varepsilon$  then
13    | exit (location of  $P_0$  is optimal);
14  else
15    DESCENT DIRECTION:
16    if  $\det \mathbf{H}_W(\mathbf{x}_0^j) < \delta$  then
17      | steepest descent  $\mathbf{d}^j = -\nabla W(\mathbf{x}_0^j)$ ;
18    else
19      | Newton's direction  $\mathbf{d}^j = -(\mathbf{H}_W^{-1} \nabla W)(\mathbf{x}_0^j)$ ;
20      |  $\cos \theta^j = -(\nabla W(\mathbf{x}_0^j))^T \mathbf{d}^j / (\|\nabla W(\mathbf{x}_0^j)\| \|\mathbf{d}^j\|)$ ;
21      | if  $\cos \theta^j < \eta$  then
22        | | steepest descent  $\mathbf{d}^j = -\nabla W(\mathbf{x}_0^j)$ ;
23    LINE SEARCH:
24    while element in the ball  $e \leq n_{\text{el}}$  do
25      | compute element objective function  $w(\mathbf{x}_0^j + \lambda^j \mathbf{d}^j)$ ;
26      |  $W(\mathbf{x}_0^j + \lambda^j \mathbf{d}^j) \leftarrow W(\mathbf{x}_0^j + \lambda^j \mathbf{d}^j) + w(\mathbf{x}_0^j + \lambda^j \mathbf{d}^j)$ ;
27    if  $W(\mathbf{x}_0^j + \lambda^j \mathbf{d}^j) - W(\mathbf{x}_0^j) \leq \frac{1}{2} \lambda^j (\nabla W(\mathbf{x}_0^j))^T \mathbf{d}^j$  then
28      | update nodal location  $\mathbf{x}_0^{j+1} = \mathbf{x}_0^j + \lambda^j \mathbf{d}^j$ , whereas  $\lambda^{j+1} = \lambda^j$ ;
29    else
30      | update step size  $\lambda^{j+1} = \frac{1}{2} \lambda^j$ , whereas  $\mathbf{x}_0^{j+1} = \mathbf{x}_0^j$ ;
31 location of  $P_0$  smoothing the ball is  $\mathbf{x}'_0 = \mathbf{x}_0^j$ ;

```

Extension to non-simplicial and/or higher-order element types with midside nodes would generally require completely different algorithms. However, a non-simplicial element can be divided into simplices which can be processed by the current procedure. For elements with midside nodes, a cheap but probably inadequate approach would be to relocate only the corner nodes of an element and to interpolate the midside nodes.

Smoothing of a surface triangle mesh by using the current algorithm is non-trivial. One method to generate a surface mesh is to regard the domain to be meshed as a parametric surface [4, 30], described by a two-dimensional parametric domain $\mathcal{D} \subset \mathbb{R}^2$ and a smooth embedding $\theta : \mathcal{D} \rightarrow \mathbb{R}^3$. Once the parametric domain has been meshed, the map θ establishes the surface mesh. The triangle mesh in the parametric space can be properly smoothed by using the current algorithm, but the resulting quality of the corresponding surface mesh is governed by θ .

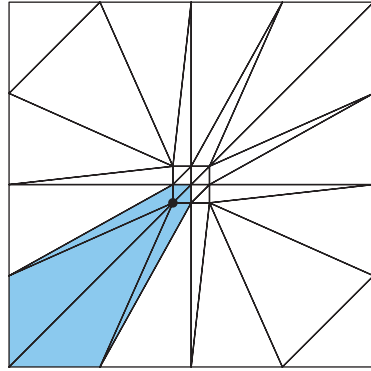
6 Numerical examples

This section presents numerical examples highlighting the applicability of the developed smoothing algorithm to different types of meshes and mesh configurations. For reasons of comparison, two additional smoothing algorithms for internal meshes have been implemented. These heuristic algorithms may replace Alg. 3 and likewise process the ball of elements associated with a single internal node. The first is based on weighted averaging and is used in the ALE method of Aymone *et al.* [14]. The second algorithm has been developed by Giuliani [15]. Its basic ingredient is an objective function whose minimum yields closed-form expressions for the coordinates of the internal node supposed to smooth the ball.

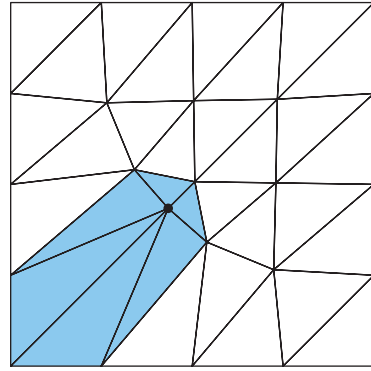
6.1 Patch tests

The example shown in Fig. 1 is a structured square patch consisting of 32 triangle elements. The best quality of the given mesh is obtained if the elements were arranged in a rising diagonals triangle pattern. In the initial configuration, however, elements are severely distorted. Merely the placement of the boundary nodes is optimal. Due to locality of the implemented smoothing algorithms, an acceptable mesh quality cannot be achieved in one step but requires several repetition loops over the balls of elements sharing a common internal node; cf. Alg. 1. However, five to ten loops are sufficient to produce an almost optimal mesh. This is independent of the particular local smoothing algorithm used for the internal mesh.

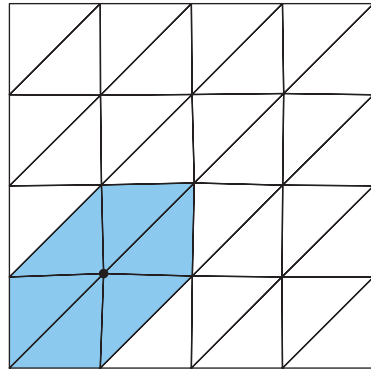
In the example, the quality improvement of the ball associated with the lower left internal node (blue zone in Fig. 1) lags behind the other after two iterations. This is a consequence of the current strategy that globally improves the mesh: all balls in the mesh are processed in a fixed order in every repetition loop. It might be



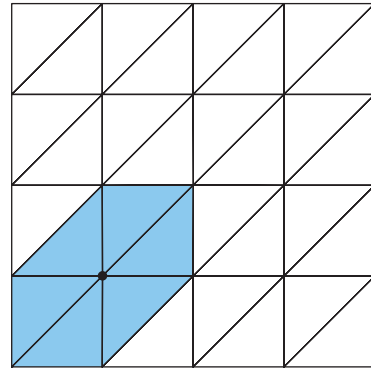
initial mesh



2 repetition loops

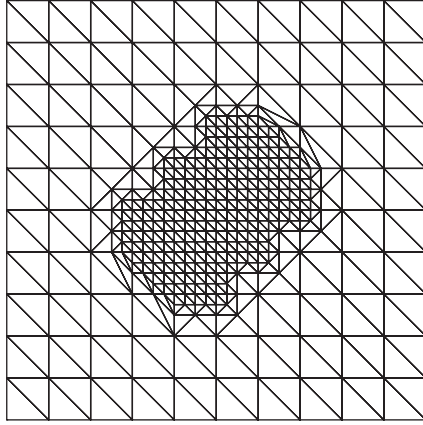


5 repetition loops

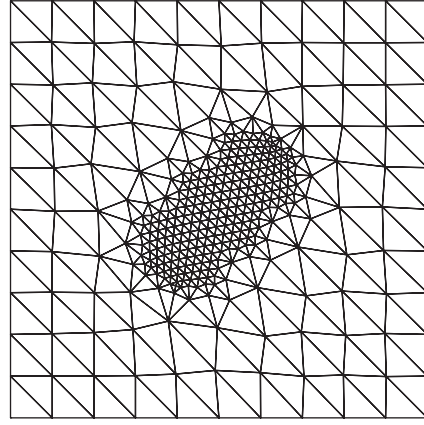


10 repetition loops

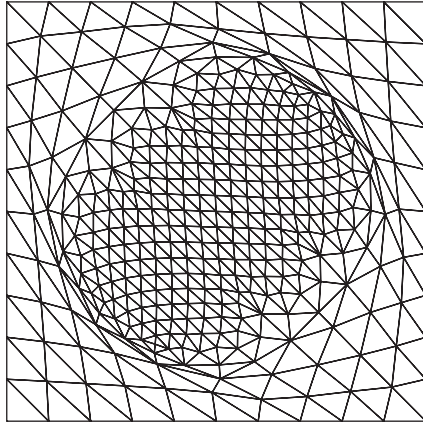
Figure 1: Investigation of the number of repetition loops required to globally improve the mesh. The blue zone indicates the ball of elements sharing the lower left internal node.



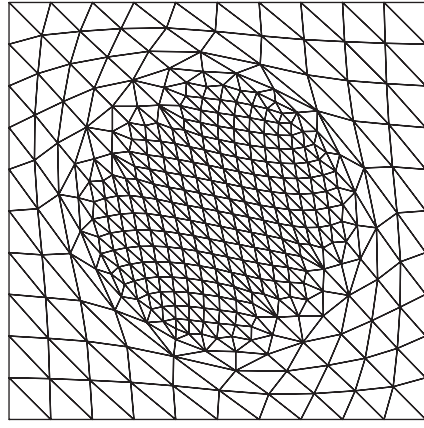
(a) initial mesh



(b) optimization-based
smoothing (OSMOT)



(c) Giuliani's smoothing
method



(d) smoothing by averaging

Figure 2: Influence of the smoothing algorithm on mesh grading after 1000 repetition loops.

more effective to process randomly picked groups of elements, but this has not been implemented yet.

The influence of the smoothing algorithm on mesh grading is investigated in the second example. Graded or anisotropic meshes made up of elements of prescribed size are often present in finite element analysis, e.g. when the computational model contains regions of secondary interest. In these cases it is important to preserve the prescribed element size. Fig. 2a shows a structured triangle mesh zone with constant density interlaced with a coarser structured mesh. The small interface zone is unstructured and contains distorted elements, whereas the structured parts of the mixed mesh are of best quality. An appropriate smoothing algorithm hence would improve the interface zone and would leave the structured zones unchanged.

It can be seen from Fig. 2b that after 1000 global loops running over the internal nodes the optimization-based algorithm (Alg. 3) results in the mesh with the highest quality. Structure is disturbed only slightly and the node density distribution resp. the size of elements is largely preserved. In contrast to that, Giuliani’s method [15] as well as smoothing by weighted averaging [14] fail the test (Fig. 2c and d, respectively). Both heuristic procedures blow up the finer mesh zone, leading to a mesh with equal-sized elements at repeated application. The quality of elements at the interface deteriorated, Giuliani’s method even caused degenerate elements. The tendency to equalize the size of elements is an undesirable feature which arises from the use of averaged geometric measures in the governing equations of the heuristic algorithms.

6.2 Non-convexly distorted mesh

A non-convexly distorted mesh is a mesh that contains stretched and/or skewed elements in the vicinity of the indented boundary, which probably has a high curvature. The automatic regularization of such a mesh at fixed connectivity is very challenging. On the other hand, problems associated with non-convexly distorted meshes constitute important benchmark problems for the implemented smoothing algorithms.

Backward extrusion is a common numerical example where non-convex regions are created when large material deformation occurs. In this initial boundary value problem a billet is loaded into a heavy walled container and then a die is moved towards the billet, so that the material is pushed through the die. Provided that the die and the container are rigid and their surfaces are rough respectively smooth, it suffices to discretize only the billet by finite elements (Fig. 3). Nodes aligned with the lower horizontal boundary are fixed in vertical direction, whereas nodes at the wall of the container are fixed in horizontal direction. The nodes located directly below the die are horizontally fixed and will be displaced in vertical direction to model the die moving downward.

Fig. 3 above shows the edges of the undeformed billet together with the deformed mesh at 30 % height reduction. The left hand side shows the results of the calculation using a heuristic scheme for mesh smoothing. Giuliani’s method [15] for internal meshes has been employed, but smoothing by weighted averaging would yield similar results. The mesh on the right hand side results from the optimization-based smoothing algorithm developed in this paper. In both calculations the simple averaging procedure summarized in Alg. 2 was chosen to smooth the boundary mesh.

The mesh quality of regions immediately under the die is comparable at 30 % height reduction. Near the lower boundary, the optimization-based algorithm produces a slightly smoother mesh. At 50 % reduction, the heavy squeezing of elements around the corner of the die cannot be avoided when using the heuristic method. The area of one element even vanishes, which inhibits convergence of the solution at con-

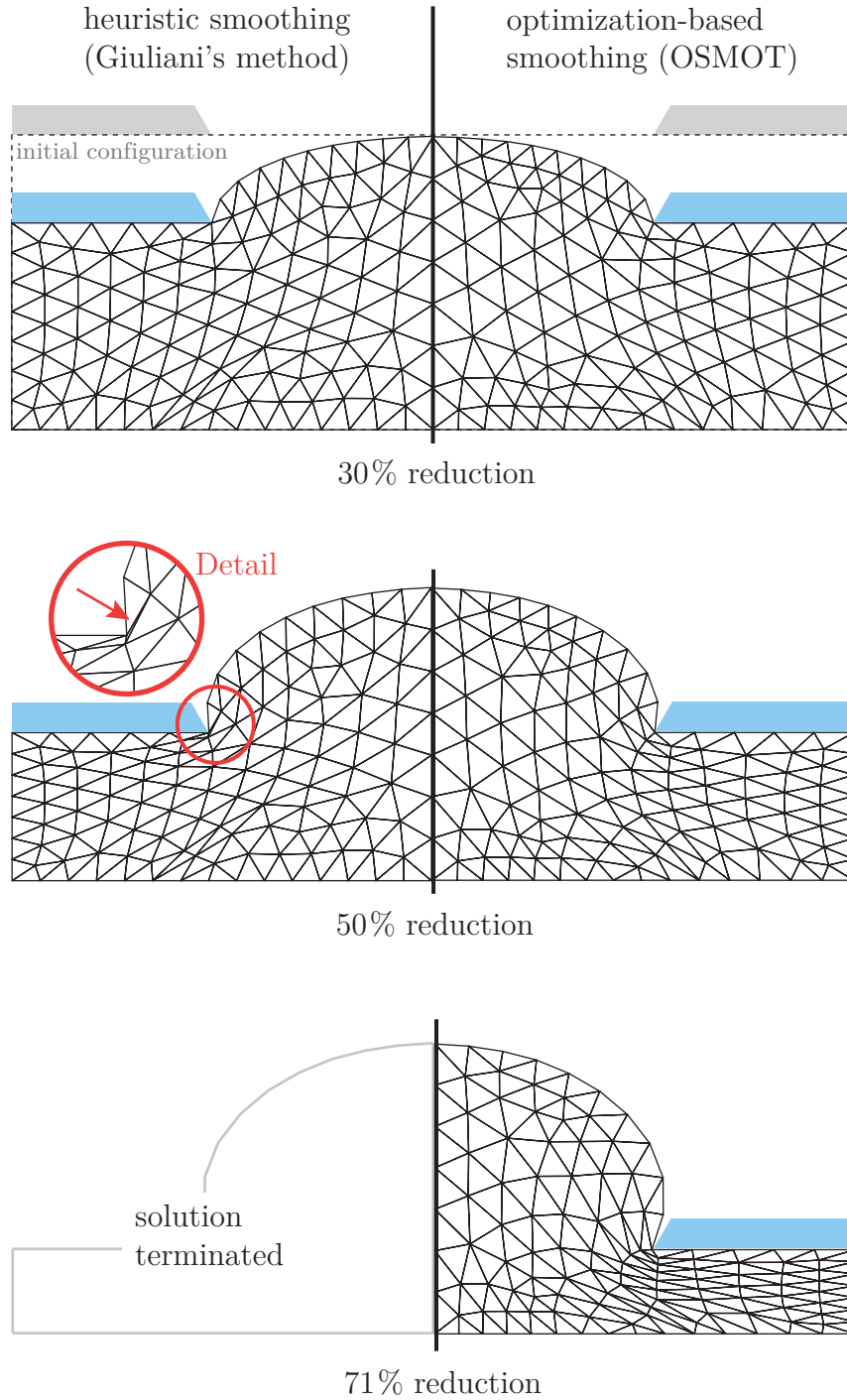


Figure 3: Comparison of an heuristic smoothing method and the developed optimization-based algorithm when applied to the numerical simulation of backward extrusion.

tinued extrusion. Compared to the heuristic method, optimization-based smoothing achieves an excellent mesh regularization. At 50 % height reduction, element squeezing is moderate, even in the non-convexly distorted region at the corner of the die. However, at continued extrusion the fixed mesh connectivity associated with smoothing algorithms limits gains of mesh quality. Calculation terminates at height reductions of more than 71 %. Only a complete remeshing would eliminate degenerate elements so as to continue solution.

6.3 Penetration of a flat-ended pile

The rigorous modeling of penetration is very challenging, especially when the behavior of the penetrated material is highly nonlinear. The final example is concerned with mesh smoothing during the ALE simulation of the penetration of a flat-ended pile into sand. It should be considered as an academic extreme example highlighting the robustness of the new smoothing algorithm when applied to large deformation problems involving indented material boundaries. Details of the ALE method and the constitutive equation used to model the mechanical behavior of sand can be found in [11].

The axisymmetric finite element model is depicted in Fig. 4a. As penetration starts from the ground surface, the initial configuration has a simple geometry. The pile is assumed rigid, its shaft is assumed perfectly smooth, and the pile base is perfectly rough (no sliding). The entire pile skin and the ground surface are modeled as a contact pair using straight segments for the sand surface and accounting for large deformation of the interface. All nodes at the lower boundary of the computational domain are fixed in vertical direction, and the nodes at the vertical boundaries are fixed in radial direction.

For a relative penetration depth of $z/D = 4.5$, where D denotes the diameter of the pile, the deformed and smoothed mesh is shown in Fig. 4b. The initially rectangular computational domain is severely deformed by indentation, resulting in a drastic increase of the perimeter-to-area ratio. Elements at the elongated boundary are stretched, i.e. the density of nodes is reduced. On the other hand, the local reduction in height of the domain below the pile base comes along with squeezing of elements. Optimization-based smoothing has indeed improved mesh quality in this example. However, it could not completely avoid element distortion because the mesh topology is fixed, that is, the mesh underneath the pile cannot “get out of the way”. Unless the computational domain would be completely remeshed the numerical model must contain a larger “stockpile” of less deformed mesh in vertical direction in order to achieve a higher mesh quality.

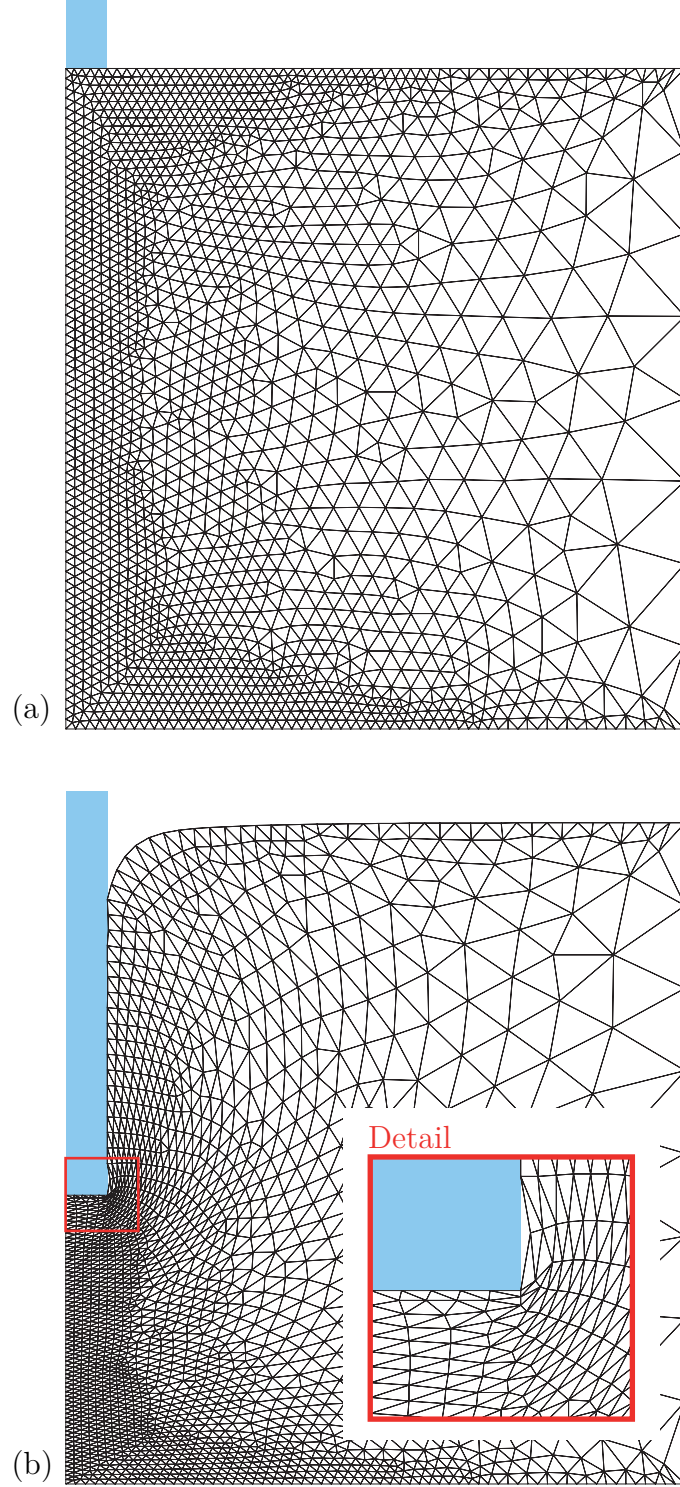


Figure 4: ALE simulation of the penetration of a flat-ended pile using the optimization-based algorithm. (a) Initial mesh, (b) deformed and smoothed mesh at a relative penetration depth of $z/D = 4.5$.

7 Conclusions

Heuristic smoothing algorithms, though they are simple and fast, are inapplicable if the meshed domain becomes non-convex with high curvature. Such situations may occur in initial mesh improvement as well as in numerical simulations of large deformation problems. In order to overcome these problems, an optimization-based smoothing algorithm for triangle meshes has been developed in this paper. The main objective was to continuously rezone the mesh in an arbitrary Lagrangian-Eulerian method for penetration problems [11].

The smoothing algorithm operates iteratively on a local level and distinguishes between boundary nodes and internal nodes. For internal nodes, an optimization procedure has been developed which processes the ball of elements enclosing a common node. It is initiated if the quality measure based on the triangle's radius ratio drops below a certain value specified by the user. A globally smoothed mesh is pursued by loops repeatedly running over the nodes of elements that fail the quality check. Numerical examples show that the overall procedure, referred to as the OS-MOT (Optimization-based SMOothing of Triangle meshes) algorithm, is efficient, extremely robust, and delivers excellent results for both structured and unstructured triangle meshes over arbitrarily shaped (i.e. convex and non-convex) domains.

During penetration, an initially convex computational domain necessarily becomes indented resp. non-convex with high curvature when material boundaries are explicitly resolved by element edges. At drastic changes of the domain's shape due to large penetration distances mesh quality improvement by smoothing can only be achieved if there is a sufficiently large stockpile of less deformed mesh. This would generally call for numerical models in which the number of finite elements becoming additionally necessary increases disproportionately with the desired penetration distance. In this case, however, it is recommended to completely remesh the computational domain or to try a different modeling technique.

Acknowledgements

The presented research work was carried out under the financial support from the German Research Foundation (DFG), grant SA 310/21-1, which is gratefully acknowledged.

A Geometric primitives of triangles

Consider a generic element $\triangle \subset \mathcal{S} = \mathbb{R}^2$ of a plane triangle mesh representing a 2-simplex. The local nodes $0, 1, 2 \in \triangle$ occupy points $\mathbf{x}_k = (x_k, y_k)^T \in \mathbb{R}^2$, with $k \in \{0, 1, 2\}$. The local connectivity of the generic triangle is predefined by choosing

the first node 0 and then assigning the numbers of the other two nodes 1 and 2 in a counter-clockwise manner such that the signed area A given by

$$2A = \det \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix} \quad (\text{A1})$$

is positive. The lengths of the edges are readily available from

$$a \stackrel{\text{def}}{=} \|\mathbf{x}_1 - \mathbf{x}_0\|, \quad b \stackrel{\text{def}}{=} \|\mathbf{x}_2 - \mathbf{x}_1\|, \quad \text{and} \quad c \stackrel{\text{def}}{=} \|\mathbf{x}_0 - \mathbf{x}_2\|, \quad (\text{A2})$$

with $\|\mathbf{x}_k - \mathbf{x}_l\| = \sqrt{(x_k - x_l)^2 + (y_k - y_l)^2}$ and $k, l \in \{0, 1, 2\}$. The following relations for triangles are well-known from undergraduate texts in geometry [31]:

$$\text{semiperimeter:} \quad s = \frac{1}{2}(a + b + c), \quad (\text{A3})$$

$$\text{Heron's formula:} \quad |A| = \sqrt{s(s-a)(s-b)(s-c)}, \quad (\text{A4})$$

$$\text{incircle radius:} \quad r = \frac{|A|}{s}, \quad (\text{A5})$$

$$\text{circumcircle radius:} \quad R = \frac{abc}{4|A|}. \quad (\text{A6})$$

The functional dependence of s, A, r , and R on $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$ is being understood.

B Gradient and Hessian for mesh optimization

B.1 General remarks

The optimization-based iterative local mesh smoothing algorithm for internal nodes (Alg. 3) requires frequent evaluation of the gradient and Hessian of the element objective function. By assuming that the numbering of the local nodes is in accordance with Section 2.1 and assuming that the locations $\mathbf{x}_1, \mathbf{x}_2$ of the local triangle nodes $1, 2 \in \triangle$ are constant during the iteration process, the currently implemented objective function for a triangle in the j -th iteration takes the form

$$w(\mathbf{x}_0^j) = \frac{R}{R_{\text{ref}}} \left(\frac{R}{r} \right)^3 = \frac{(abc)^4 s^3}{4^4 R_{\text{ref}} A^7}, \quad (\text{B1})$$

where (A5) and (A6) have been used, and the functional dependence of a, b, c, s, A, r, R on $(\mathbf{x}_0^j, \mathbf{x}_1, \mathbf{x}_2)$ is being understood. By dropping the superscribed j indicating the iteration step in what follows, the gradient and Hessian of $w(\mathbf{x}_0)$ in \mathbb{R}^2 are the component matrices given by

$$\nabla w(\mathbf{x}_0) = \begin{pmatrix} \frac{\partial w}{\partial x_0} \\ \frac{\partial w}{\partial y_0} \end{pmatrix} \quad \text{and} \quad \mathbf{H}_w(\mathbf{x}_0) = \begin{pmatrix} \frac{\partial^2 w}{\partial x_0^2} & \frac{\partial^2 w}{\partial x_0 \partial y_0} \\ \frac{\partial^2 w}{\partial y_0 \partial x_0} & \frac{\partial^2 w}{\partial y_0^2} \end{pmatrix}, \quad (\text{B2})$$

respectively. The geometric primitives of triangles provided in A enable the straightforward calculation of the components of $\nabla w(\mathbf{x}_0)$ and $\mathbf{H}_w(\mathbf{x}_0)$, see below. Note that the derivatives of $b = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ with respect to \mathbf{x}_0 , that is, $\frac{\partial b}{\partial x_0}$, $\frac{\partial b}{\partial y_0}$, $\frac{\partial^2 b}{\partial x_0 \partial y_0}$, etc., identically vanish.

B.2 First derivatives of objective function

$$\frac{\partial w}{\partial x_0} = \frac{1}{4^4 R_{\text{ref}}} \left(\frac{s^3}{A^7} \frac{\partial}{\partial x_0} (abc)^4 + \frac{(abc)^4}{A^7} \frac{\partial}{\partial x_0} s^3 + (abc)^4 s^3 \frac{\partial}{\partial x_0} A^{-7} \right) \quad (\text{B3})$$

$$\frac{\partial w}{\partial y_0} = \frac{1}{4^4 R_{\text{ref}}} \left(\frac{s^3}{A^7} \frac{\partial}{\partial y_0} (abc)^4 + \frac{(abc)^4}{A^7} \frac{\partial}{\partial y_0} s^3 + (abc)^4 s^3 \frac{\partial}{\partial y_0} A^{-7} \right) \quad (\text{B4})$$

B.2.1 Extensions

$$\frac{\partial}{\partial x_0} (abc)^4 = 4b(abc)^3 (cC_{1x} + aC_{2x}) \quad (\text{B5})$$

$$\frac{\partial}{\partial y_0} (abc)^4 = 4b(abc)^3 (cC_{1y} + aC_{2y}) \quad (\text{B6})$$

$$\frac{\partial}{\partial x_0} s^3 = 3s^2 C_{3x} \quad (\text{B7})$$

$$\frac{\partial}{\partial y_0} s^3 = 3s^2 C_{3y} \quad (\text{B8})$$

$$\begin{aligned} \frac{\partial}{\partial x_0} A^{-7} = & -\frac{7}{2A^9} \left[(2s - b)(s - a)(s - c)C_{3x} \right. \\ & \left. + s(s - b)(s - c)C_{4x} + s(s - a)(s - b)C_{5x} \right] \end{aligned} \quad (\text{B9})$$

$$\begin{aligned} \frac{\partial}{\partial y_0} A^{-7} = & -\frac{7}{2A^9} \left[(2s - b)(s - a)(s - c)C_{3y} \right. \\ & \left. + s(s - b)(s - c)C_{4y} + s(s - a)(s - b)C_{5y} \right] \end{aligned} \quad (\text{B10})$$

$$\frac{\partial A}{\partial x_0} = -\frac{A^8}{7} \frac{\partial}{\partial x_0} A^{-7} \quad (\text{B11})$$

$$\frac{\partial A}{\partial y_0} = -\frac{A^8}{7} \frac{\partial}{\partial y_0} A^{-7} \quad (\text{B12})$$

B.2.2 Abbreviations

$$C_{1x} \stackrel{\text{def}}{=} \frac{\partial a}{\partial x_0} = \frac{\partial}{\partial x_0} \left((x_1 - x_0)^2 + (y_1 - y_0)^2 \right)^{\frac{1}{2}} = -\frac{x_1 - x_0}{a} \quad (\text{B13})$$

$$C_{1y} \stackrel{\text{def}}{=} \frac{\partial a}{\partial y_0} = \frac{\partial}{\partial y_0} \left((x_1 - x_0)^2 + (y_1 - y_0)^2 \right)^{\frac{1}{2}} = -\frac{y_1 - y_0}{a} \quad (\text{B14})$$

$$C_{2x} \stackrel{\text{def}}{=} \frac{\partial c}{\partial x_0} = \frac{\partial}{\partial x_0} \left((x_0 - x_2)^2 + (y_0 - y_2)^2 \right)^{\frac{1}{2}} = \frac{x_0 - x_2}{c} \quad (\text{B15})$$

$$C_{2y} \stackrel{\text{def}}{=} \frac{\partial c}{\partial y_0} = \frac{\partial}{\partial y_0} \left((x_0 - x_2)^2 + (y_0 - y_2)^2 \right)^{\frac{1}{2}} = \frac{y_0 - y_2}{c} \quad (\text{B16})$$

$$C_{3x} \stackrel{\text{def}}{=} \frac{\partial s}{\partial x_0} = \frac{1}{2} \left(\frac{\partial a}{\partial x_0} + \frac{\partial b}{\partial x_0} + \frac{\partial c}{\partial x_0} \right) = \frac{1}{2} (C_{1x} + C_{2x}) \quad (\text{B17})$$

$$C_{3y} \stackrel{\text{def}}{=} \frac{\partial s}{\partial y_0} = \frac{1}{2} \left(\frac{\partial a}{\partial y_0} + \frac{\partial b}{\partial y_0} + \frac{\partial c}{\partial y_0} \right) = \frac{1}{2} (C_{1y} + C_{2y}) \quad (\text{B18})$$

$$C_{4x} \stackrel{\text{def}}{=} \frac{\partial(s-a)}{\partial x_0} = \frac{\partial s}{\partial x_0} - \frac{\partial a}{\partial x_0} = C_{3x} - C_{1x} \quad (\text{B19})$$

$$C_{4y} \stackrel{\text{def}}{=} \frac{\partial(s-a)}{\partial y_0} = \frac{\partial s}{\partial y_0} - \frac{\partial a}{\partial y_0} = C_{3y} - C_{1y} \quad (\text{B20})$$

$$C_{5x} \stackrel{\text{def}}{=} \frac{\partial(s-c)}{\partial x_0} = \frac{\partial s}{\partial x_0} - \frac{\partial c}{\partial x_0} = C_{3x} - C_{2x} \quad (\text{B21})$$

$$C_{5y} \stackrel{\text{def}}{=} \frac{\partial(s-c)}{\partial y_0} = \frac{\partial s}{\partial y_0} - \frac{\partial c}{\partial y_0} = C_{3y} - C_{2y} \quad (\text{B22})$$

B.3 Second derivatives of objective function

$$\begin{aligned} \frac{\partial^2 w}{\partial x_0^2} = & \frac{1}{4^4 R_{\text{ref}}} \left(\frac{s^3}{A^7} \frac{\partial^2}{\partial x_0^2} (abc)^4 + \frac{(abc)^4}{A^7} \frac{\partial^2}{\partial x_0^2} s^3 + s^3 (abc)^4 \frac{\partial^2}{\partial x_0^2} A^{-7} \right. \\ & + \frac{2}{A^7} \frac{\partial}{\partial x_0} (abc)^4 \frac{\partial}{\partial x_0} s^3 + 2s^3 \frac{\partial}{\partial x_0} (abc)^4 \frac{\partial}{\partial x_0} A^{-7} \\ & \left. + 2(abc)^4 \frac{\partial}{\partial x_0} s^3 \frac{\partial}{\partial x_0} A^{-7} \right) \end{aligned} \quad (\text{B23})$$

$$\begin{aligned} \frac{\partial^2 w}{\partial y_0^2} = & \frac{1}{4^4 R_{\text{ref}}} \left(\frac{s^3}{A^7} \frac{\partial^2}{\partial y_0^2} (abc)^4 + \frac{(abc)^4}{A^7} \frac{\partial^2}{\partial y_0^2} s^3 + s^3 (abc)^4 \frac{\partial^2}{\partial y_0^2} A^{-7} \right. \\ & + \frac{2}{A^7} \frac{\partial}{\partial y_0} (abc)^4 \frac{\partial}{\partial y_0} s^3 + 2s^3 \frac{\partial}{\partial y_0} (abc)^4 \frac{\partial}{\partial y_0} A^{-7} \\ & \left. + 2(abc)^4 \frac{\partial}{\partial y_0} s^3 \frac{\partial}{\partial y_0} A^{-7} \right) \end{aligned} \quad (\text{B24})$$

$$\begin{aligned} \frac{\partial^2 w}{\partial x_0 \partial y_0} = & \frac{1}{4^4} \left(\frac{s^3}{A^7} \frac{\partial^2}{\partial x_0 \partial y_0} (abc)^4 + \frac{1}{A^7} \frac{\partial}{\partial x_0} (abc)^4 \frac{\partial}{\partial y_0} s^3 \right. \\ & + s^3 \frac{\partial}{\partial x_0} (abc)^4 \frac{\partial}{\partial y_0} A^{-7} + \frac{1}{A^7} \frac{\partial}{\partial y_0} (abc)^4 \frac{\partial}{\partial x_0} s^3 \\ & \left. + \frac{(abc)^4}{A^7} \frac{\partial^2}{\partial x_0 \partial y_0} s^3 + (abc)^4 \frac{\partial}{\partial x_0} s^3 \frac{\partial}{\partial y_0} A^{-7} \right) \end{aligned}$$

$$\begin{aligned}
& + s^3 \frac{\partial}{\partial y_0} (abc)^4 \frac{\partial}{\partial x_0} A^{-7} + (abc)^4 \frac{\partial}{\partial y_0} s^3 \frac{\partial}{\partial x_0} A^{-7} \\
& + s^3 (abc)^4 \frac{\partial^2}{\partial x_0 \partial y_0} A^{-7} \Big) = \frac{\partial^2 w}{\partial y_0 \partial x_0}
\end{aligned} \tag{B25}$$

B.3.1 Extensions

$$\begin{aligned}
\frac{\partial^2}{\partial^2 x_0} (abc)^4 &= 12b(abc)^2 (cC_{1x} + aC_{2x})^2 \\
&+ 4b(abc)^3 (cD_{1x} + aD_{2x} + 2C_{1x}C_{2x})
\end{aligned} \tag{B26}$$

$$\begin{aligned}
\frac{\partial^2}{\partial^2 y_0} (abc)^4 &= 12b(abc)^2 (cC_{1y} + aC_{2y})^2 \\
&+ 4b(abc)^3 (cD_{1y} + aD_{2y} + 2C_{1y}C_{2y})
\end{aligned} \tag{B27}$$

$$\begin{aligned}
\frac{\partial^2}{\partial x_0 \partial y_0} (abc)^4 &= 12b(abc)^2 (cC_{1x} + aC_{2x})(cC_{1y} + aC_{2y}) \\
&+ 4b(abc)^3 (cE_1 + C_{1x}C_{2y} + C_{1y}C_{2x} + aE_2)
\end{aligned} \tag{B28}$$

$$\frac{\partial^2}{\partial x_0^2} s^3 = 6sC_{3x}^2 + 3s^2 D_{3x} \tag{B29}$$

$$\frac{\partial^2}{\partial y_0^2} s^3 = 6sC_{3y}^2 + 3s^2 D_{3y} \tag{B30}$$

$$\frac{\partial^2}{\partial x_0 \partial y_0} s^3 = 6sC_{3x}C_{3y} + 3s^2 E_3 \tag{B31}$$

$$\begin{aligned}
\frac{\partial^2}{\partial x_0^2} A^{-7} &= \frac{63}{A^9} \left(\frac{\partial A}{\partial x_0} \right)^2 \\
&- \frac{7}{2A^9} \left[(s-a) \{ (s-b)(sD_{5x} + (s-c)D_{3x} + 2C_{3x}C_{5x}) \right. \\
&+ (s-c)(sD_{3x} + 2C_{3x}^2) + 2sC_{3x}C_{5x} \} \\
&+ (s-b) \{ (s-c)(sD_{4x} + 2C_{3x}C_{4x}) + 2sC_{4x}C_{5x} \} \\
&\left. + 2s(s-c)C_{3x}C_{4x} \right]
\end{aligned} \tag{B32}$$

$$\begin{aligned}
\frac{\partial^2}{\partial y_0^2} A^{-7} &= \frac{63}{A^9} \left(\frac{\partial A}{\partial y_0} \right)^2 \\
&- \frac{7}{2A^9} \left[(s-a) \{ (s-b)(sD_{5y} + (s-c)D_{3y} + 2C_{3y}C_{5y}) \right. \\
&+ (s-c)(sD_{3y} + 2C_{3y}^2) + 2sC_{3y}C_{5y} \} \\
&+ (s-b) \{ (s-c)(sD_{4y} + 2C_{3y}C_{4y}) + 2sC_{4y}C_{5y} \} \\
&\left. + 2s(s-c)C_{3y}C_{4y} \right]
\end{aligned} \tag{B33}$$

$$\frac{\partial^2}{\partial x_0 \partial y_0} A^{-7} = \frac{63}{A^9} \frac{\partial A}{\partial y_0} \frac{\partial A}{\partial x_0}$$

$$\begin{aligned}
& -\frac{7}{2A^9} \left[(s-a) \{ (s-b)(C_{3x}C_{5y} + (s-c)E_3 + C_{5x}C_{3y} + sE_5) \right. \\
& + (s-c)(2C_{3x}C_{3y} + sE_3) + s(C_{5x}C_{3y} + C_{3x}C_{5y}) \} \\
& + s(s-c)(C_{3x}C_{4y} + C_{4x}C_{3y}) + (s-b) \{ s(C_{4x}C_{5y} + C_{5x}C_{4y}) \\
& \left. + (s-c)(C_{3x}C_{4y} + C_{4x}C_{3y} + sE_4) \} \right] \quad (B34)
\end{aligned}$$

B.3.2 Abbreviations

$$D_{1x} \stackrel{\text{def}}{=} \frac{\partial^2 a}{\partial x_0^2} = \frac{\partial}{\partial x_0} \left(-\frac{x_1 - x_0}{a} \right) = \frac{x_1 - x_0}{a^2} C_{1x} + \frac{1}{a} \quad (B35)$$

$$D_{1y} \stackrel{\text{def}}{=} \frac{\partial^2 a}{\partial y_0^2} = \frac{\partial}{\partial y_0} \left(-\frac{y_1 - y_0}{a} \right) = \frac{y_1 - y_0}{a^2} C_{1y} + \frac{1}{a} \quad (B36)$$

$$D_{2x} \stackrel{\text{def}}{=} \frac{\partial^2 c}{\partial x_0^2} = -\frac{x_0 - x_2}{c^2} C_{2x} + \frac{1}{c} \quad (B37)$$

$$D_{2y} \stackrel{\text{def}}{=} \frac{\partial^2 c}{\partial y_0^2} = -\frac{y_0 - y_2}{c^2} C_{2y} + \frac{1}{c} \quad (B38)$$

$$D_{3x} \stackrel{\text{def}}{=} \frac{\partial^2 s}{\partial x_0^2} = \frac{\partial}{\partial x_0} \left(\frac{\partial}{\partial x_0} \frac{a+b+c}{2} \right) = \frac{1}{2} (D_{1x} + D_{2x}) \quad (B39)$$

$$D_{3y} \stackrel{\text{def}}{=} \frac{\partial^2 s}{\partial y_0^2} = \frac{\partial}{\partial y_0} \left(\frac{\partial}{\partial y_0} \frac{a+b+c}{2} \right) = \frac{1}{2} (D_{1y} + D_{2y}) \quad (B40)$$

$$D_{4x} \stackrel{\text{def}}{=} \frac{\partial^2 (s-a)}{\partial x_0^2} = \frac{\partial^2 s}{\partial x_0^2} - \frac{\partial^2 a}{\partial x_0^2} = D_{3x} - D_{1x} \quad (B41)$$

$$D_{4y} \stackrel{\text{def}}{=} \frac{\partial^2 (s-a)}{\partial y_0^2} = \frac{\partial^2 s}{\partial y_0^2} - \frac{\partial^2 a}{\partial y_0^2} = D_{3y} - D_{1y} \quad (B42)$$

$$D_{5x} \stackrel{\text{def}}{=} \frac{\partial^2 (s-c)}{\partial x_0^2} = \frac{\partial^2 s}{\partial x_0^2} - \frac{\partial^2 c}{\partial x_0^2} = D_{3x} - D_{2x} \quad (B43)$$

$$D_{5y} \stackrel{\text{def}}{=} \frac{\partial^2 (s-c)}{\partial y_0^2} = \frac{\partial^2 s}{\partial y_0^2} - \frac{\partial^2 c}{\partial y_0^2} = D_{3y} - D_{2y} \quad (B44)$$

$$E_1 \stackrel{\text{def}}{=} \frac{\partial^2 a}{\partial x_0 \partial y_0} = -\frac{\partial}{\partial y_0} \left(\frac{x_1 - x_0}{a} \right) = -\frac{(x_1 - x_0)(y_1 - y_0)}{a^3} \quad (B45)$$

$$E_2 \stackrel{\text{def}}{=} \frac{\partial^2 c}{\partial x_0 \partial y_0} = \frac{\partial}{\partial y_0} \left(\frac{x_0 - x_2}{c} \right) = -\frac{(x_0 - x_2)(y_0 - y_2)}{c^3} \quad (B46)$$

$$E_3 \stackrel{\text{def}}{=} \frac{\partial^2 s}{\partial x_0 \partial y_0} = \frac{1}{2} \left(\frac{\partial^2 a}{\partial x_0 \partial y_0} + \frac{\partial^2 b}{\partial x_0 \partial y_0} + \frac{\partial^2 c}{\partial x_0 \partial y_0} \right) = \frac{1}{2} (E_1 + E_2) \quad (B47)$$

$$E_4 \stackrel{\text{def}}{=} \frac{\partial^2 (s-a)}{\partial x_0 \partial y_0} = \frac{\partial^2 s}{\partial x_0 \partial y_0} - \frac{\partial^2 a}{\partial x_0 \partial y_0} = E_3 - E_1 \quad (B48)$$

$$E_5 \stackrel{\text{def}}{=} \frac{\partial^2 (s-c)}{\partial x_0 \partial y_0} = \frac{\partial^2 s}{\partial x_0 \partial y_0} - \frac{\partial^2 c}{\partial x_0 \partial y_0} = E_3 - E_2 \quad (B49)$$

References

- [1] A. Liu and B. Joe. Relationship between tetrahedron shape measures. *BIT*, 34:268–287, 1994.
- [2] D. A. Field. Qualitative measures for initial meshes. *International Journal for Numerical Methods in Engineering*, 47:887–906, 2000.
- [3] P. M. Knupp. Algebraic mesh quality metrics. *SIAM Journal on Scientific Computing*, 23(1):193–218, 2001.
- [4] P. J. Frey and P.-L. George. *Mesh Generation: Application to Finite Elements*. HERMES Science Europe Ltd., Oxford, UK, 2000.
- [5] C. W. Hirt, A. A. Amsden, and J. L. Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of Computational Physics*, 14:227–253, 1974.
- [6] T. J. R. Hughes, W. K. Liu, and T. K. Zimmermann. Lagrangian-Eulerian finite element formulation for incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 29:329–349, 1981.
- [7] D. J. Benson. An efficient, accurate, simple ALE method for nonlinear finite element programs. *Computer Methods in Applied Mechanics and Engineering*, 72(3):305–350, 1989.
- [8] M. Nazem, D. Sheng, J. P. Carter, and S. W. Sloan. Arbitrary Lagrangian-Eulerian method for large-strain consolidation problems. *International Journal for Numerical and Analytical Methods in Geomechanics*, 32(9):1023–1050, 2008.
- [9] S. A. Savidis, D. Aubram, and F. Rackwitz. Arbitrary Lagrangian-Eulerian finite element formulation for geotechnical construction processes. *Journal of Theoretical and Applied Mechanics*, 38(1-2):165–194, 2008.
- [10] D. Aubram, F. Rackwitz, and S. A. Savidis. An ALE finite element method for cohesionless soil at large strains: Computational aspects and applications. In T. Benz and S. Nordal, editors, *Proceedings 7th European Conference on Numerical Methods in Geotechnical Engineering (NUMGE 2010)*, pages 245–250. CRC Press, London, 2010.
- [11] D. Aubram. *An Arbitrary Lagrangian-Eulerian Method for Penetration into Sand at Finite Deformation*. Number 62 in Veröffentlichungen des Grundbauinstitutes der Technischen Universität Berlin. Shaker Verlag, Aachen, 2013. <http://opus4.kobv.de/opus4-tuberlin/frontdoor/index/index/docId/4755>.
- [12] D. J. Benson. Computational methods in Lagrangian and Eulerian hydrocodes. *Computer Methods in Applied Mechanics and Engineering*, 99(2-3):235–394, 1992.

- [13] J. Donea, A. Huerta, J.-Ph. Ponthot, and A. Rodríguez-Ferran. *Arbitrary Lagrangian-Eulerian Methods*, volume 1 of *Encyclopedia of Computational Mechanics*, chapter 14. John Wiley & Sons, Ltd., 2004.
- [14] J. L. F. Aymone, E. Bittencourt, and G. J. Creus. Simulation of 3d metal-forming using an arbitrary Lagrangian-Eulerian finite element method. *Journal of Materials Processing Technology*, 110:218–232, 2001.
- [15] S. Giuliani. An algorithm for continuous rezoning of the hydrodynamic grid in arbitrary Lagrangian-Eulerian computer codes. *Nuclear Engineering and Design*, 72:205–212, 1982.
- [16] R. Löhner, K. Morgan, and O. C. Zienkiewicz. Adaptive grid refinement for the compressible Euler equations. In I. Babuška, O. C. Zienkiewicz, J. Gago, and E. R. A. Oliveira, editors, *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, pages 281–297. John Wiley & Sons, Ltd., 1986.
- [17] W. D. Barfield. An optimal mesh generator for Lagrangian hydrodynamic calculations in two space dimensions. *Journal of Computational Physics*, 6:417–429, 1970.
- [18] J. U. Brackbill and J. S. Saltzman. Adaptive zoning for singular problems in two dimensions. *Journal of Computational Physics*, 46:342–368, 1982.
- [19] S. R. Kennon and G. S. Dulikravich. Generation of computational grids using optimization. *AIAA Journal*, 24(7):1069–1073, 1986.
- [20] E. A. Dari and G. C. Buscaglia. Mesh optimization: How to obtain good unstructured 3d finite element meshes with not-so-good mesh generators. *Structural and Multidisciplinary Optimization*, 8:181–188, 1993.
- [21] P. D. Zavattieri, E. A. Dari, and G. C. Buscaglia. Optimization strategies in unstructured mesh generation. *International Journal for Numerical Methods in Engineering*, 39:2055–2071, 1996.
- [22] M. S. Joun and M. C. Lee. Quadrilateral finite-element generation and mesh quality control for metal forming simulation. *International Journal for Numerical Methods in Engineering*, 40:4059–4075, 1997.
- [23] H. Braess and P. Wriggers. Arbitrary Lagrangian Eulerian finite element analysis of free surface flow. *Computer Methods in Applied Mechanics and Engineering*, 190:95–109, 2000.
- [24] P. M. Knupp, L. G. Margolin, and M. Shashkov. Reference Jacobian optimization-based rezone strategies for arbitrary Lagrangian Eulerian methods. *Journal of Computational Physics*, 176:93–128, 2002.
- [25] V. Dyadechko, R. Garimella, and M. Shashkov. Reference Jacobian rezoning strategy for arbitrary Lagrangian-Eulerian methods on polyhedral grids. Report LA-UR-05-8159, Los Alamos National Laboratory, Los Alamos, USA, 2005.

- [26] W. Sun and Y.-X. Yuan. *Optimization Theory and Methods - Nonlinear Programming*. Springer Science+Business Media, LLC, 2006.
- [27] D. G. Luenberger and Y. Ye. *Linear and Nonlinear Programming*. Springer Science+Business Media, LLC, 3rd edition, 2008.
- [28] A. A. Goldstein and J. F. Price. An effective algorithm for minimization. *Numerische Mathematik*, 10:184–189, 1967.
- [29] L. Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Computational Mechanics*, 16:1–3, 1966.
- [30] P. L. George, H. Borouchaki, P. J. Frey, P. Laug, and E. Saltel. *Mesh Generation and Mesh Adaptivity: Theory and Techniques*, volume 1 of *Encyclopedia of Computational Mechanics*, chapter 17. John Wiley & Sons, Ltd., 2004.
- [31] I. N. Bronstein, K. A. Semendjajew, G. Musiol, and H. Mühlig. *Handbook of Mathematics*. Springer-Verlag Berlin Heidelberg, 5th edition, 2007.